Hacking y Pentesting con Python

Módulo 7: Rutinas de Red Teaming con Python.

eip ESCUELA INTERNACIONAL POSGRADOS

ÍNDICE

Módulo 7: Rutinas de Red Teaming con Python.

PRESENTACIÓN Y OBJETIVOS

En este módulo se exponen algunas de las técnicas que se suelen aplicar en campañas de Red Team y cómo dichas técnicas se pueden automatizar por medio de scripts desarrollados en Python.

Objetivos

- Entender el funcionamiento de la red TOR para salvaguardar el anonimato de los usuarios y cómo puede ser utilizada para ejecutar ataques contra objetivos en internet.
- Enseñar rutinas de espionaje sobre sistemas comprometidos utilizando Python.
- Enseñar cómo automátizar procesos de post-explotación en sistemas Windows y Linux.

Red Team con Python.

1. Integración Python y TOR

Existen varias alternativas a la hora de navegar por Internet o acceder a diferentes tipos de servicios de forma anónima. Sin embargo, la solución más conocida y potente en este amplio abanico de herramientas es TOR (The Onion Router).

TOR es una red cuyo funcionamiento se basa en el cifrado y envío de paquetes de datos que viajan entre tres repetidores que conforman lo que conoce como un circuito. Cuando un cliente desea conectarse a TOR, intenta descargar el último descriptor válido emitido por las autoridades de directorio y posteriormente procede a crear un circuito que estará compuesto por un repetidor de entrada, un repetidor intermedio y finalmente, un repetidor de salida.

Para el cifrado de la información se utiliza un mecanismo de clave asimétrica, con lo cual, el cliente se encarga de solicitar la clave pública de cada uno de los repetidores que conforman su circuito y cifra todos sus paquetes de datos utilizando cada una de las claves públicas de los repetidores.

Este funcionamiento, es el motivo por el cual el logo de TOR es una cebolla, ya que en la medida que un paquete de datos llega a un repetidor del circuito, dicho repetidor utiliza su clave privada para descifrar una capa del paquete.

Cuando el paquete de datos llega al repetidor final, que es conocido también como nodo de salida, éste utiliza su clave privada para remover la última capa de cifrado del paquete, con lo cual, el nodo de salida es capaz de obtener el paquete de datos en texto plano y saber exactamente cuál es el destino.

Si el cliente no ha utilizado un mecanismo de cifrado adicional, evidentemente un nodo de salida malicioso puede analizar los paquetes de datos que recibe y realizar algún tipo de correlación o análisis estadístico que le permita conocer el origen real del paquete.

No obstante, para realizar un ataque de esas características, el atacante deberá contar con suficientes repetidores en la red de TOR y una potencia de cómputo considerable, ya que la selección y creación de circuitos la realizan los clientes de TOR de forma aleatoria y periódica. Sin embargo, en el caso de ciertas entidades gubernamentales, este hecho no supone una gran dificultad dados los recursos que tienen a su disposición.

1.1. Librería STEM para controlar una instancia de TOR desde Python.

Stem es una librería que actúa como un cliente de cualquier instancia de TOR. Su funcionamiento se basa en el uso del protocolo de control de TOR y las primitivas que tiene disponibles dicho protocolo. Permite construir aplicaciones que funcionen de un modo similar a la utilidad "ARM" ya que permite consultar y controlar todas las características de un proceso TOR local. Por otro lado, permite ejecutar otras rutinas interesantes sobre la instancia de TOR, tales como consultar los descriptores almacenados en las autoridades de directorio y posteriormente descargarlos.

El único requisito para que funcione correctamente, es que la instancia de TOR tenga establecida la propiedad "ControlPort". En Stem, la clase stem.control.Controller permite el intercambio de información entre una rutina en Python y el cliente de TOR.

```
>>> from stem.control import Controller
>>> controller = Controller.from_port(port = 9051)
>>> controller.authenticate("password1")
>>> print(controller.get_info("traffic/read"))
16701490
>>> print(controller.get_info("traffic/written"))
1160140
>>> print(controller.get_info("process/pid"))
15181
>>>
```

Figura 1.1: Uso básico de la librería Stem

En la imagen anterior se puede comprobar el uso básico. Se puede apreciar la conexión y autenticación con la instancia de TOR.

Otra característica que puede ser útil para un pentester o atacante, es la posibilidad de iniciar de forma programática una instancia de TOR, en tal caso, es necesario incluir las mismas propiedades que soporta el fichero de configuración "torrc".

El script **7-starttor.py** se encarga precisamente de realizar dicha tarea. No obstante hay que tener en cuenta que se debe indicar de forma explícita la ubicación del binario "tor". Sin esto, la librería buscará en el PATH del sistema y si no encuentra el ejecutable correspondiente se producirá un error.

Además de lo anterior, la librería también permite obtener un listado de los nodos que hacen parte de la red. Esto permite analizar nodos con versiones concretas de TOR y detalles que pueden ser interesantes como la información de contacto del nodo o su dirección IP pública.

El script **7-stemdescriptors.py** enseña cómo consultar los descriptores de TOR públicos y filtra por aquellos nodos cuya versión de TOR es inferior a la 0.4.0. Evidentemente, se puede cambiar dicho valor con el objetivo de buscar otras versiones más antiguas.

Importante

Como ocurre con muchas otras librerías en Python para pentesting, es necesario que la herramienta en cuestión se encuentre instalada en el sistema. En este caso, si se dese utilizar Stem, es fundamental que la instancia de TOR se encuentre instalada y que el binario correspondiente se encuentre ubicado en la variable PATH del sistema.

1.2. Peticiones HTTP pasando por TOR y Python.

Cuando se inicia el cliente de TOR, automáticamente el software intenta leer el fichero “torrc”, el cual a su vez contiene varias propiedades de usos muy variados, como por ejemplo la propiedad “SocksPort” que define el puerto por el que se va a iniciar un proxy del tipo socks, el cual es frecuentemente utilizado desde un navegador para visitar sitios en Internet o en la deep web de forma anónima.

No obstante, este proxy también puede utilizarse directamente desde cualquier otro programa, como por ejemplo un script en Python. Poder utilizar rutinas de código que se conecten por medio de un circuito de TOR tiene varias ventajas desde el punto de vista de un atacante, especialmente con lo referente a la ejecución de pruebas de penetración sobre aplicaciones o servidores web de forma anónima.

Para conseguir dicho objetivo, es posible utilizar algunas librerías que permiten crear conexiones por medio de un proxy socks como por ejemplo requests y socksipy. El script 7-requeststor.py enseña el uso de la librería “requests”, en donde es posible indicar un servidor proxy SOCKS para enrutar las peticiones.

2. Rutinas de Espionaje

Cuando un atacante o grupo de ciberdelincuentes consiguen romper el perímetro de seguridad de su objetivo, una de las primeras cosas que intentarán hacer es elevar sus privilegios, recopilar toda la información sensible que sea posible y establecer programas que permitan espiar de la forma más sigilosa posible todas las actividades desempeñadas por la víctima.

En este sentido, contar con herramientas que permitan realizar capturas de pantalla y registrar la actividad del teclado es primordial. Realizar estas tareas en Python no es demasiado complicado, pero requiere el uso de algunas librerías adicionales que no se encuentran incluidas en el intérprete.

A continuación se explica cuáles son dichas librerías y cómo utilizarlas para crear un programa que permita espiar las actividades de un usuario. Hay que tener en cuenta que los scripts que se ejecutan en Windows dependen del intérprete de Python, lo cual no siempre será un requisito que se cumpla en un sistema comprometido, en esos casos un atacante puede utilizar PyInstaller, Py2Exe o cxfreeze para convertir scripts en Python en formatos compatibles con los sistemas atacados.

2.1. Keylogger en Windows.

Es posible crear keylogger básico con Python, el cual se encargará de registrar todas las teclas pulsadas por la víctima y enviar dichos eventos del teclado a una máquina remota que evidentemente, estará controlada por el atacante. Para ello se puede utilizar las librerías PyWin (http://sourceforge.net/projects/pywin32/) y PyWinhook (https://www.lfd.uci.edu/~gohlke/pythonlibs/#pywinhook), las cuales solamente se encuentran disponibles en sistemas Windows.

Dichas librerías contienen varios módulos y funciones que permiten enganchar los principales eventos de entrada en el sistema, como por ejemplo eventos del teclado o del ratón.

El script **7-keyloggerWindows.py** es un ejemplo básico del uso de dichas librerías, en el cual se registrarán las teclas pulsadas por el usuario y se enviará dicho registro a una máquina controlada por el atacante por medio de una conexión TCP plana.

Es importante tener en cuenta que antes de ejecutar el script anterior se debe abrir el puerto indicado en la máquina del atacante, de esta forma se podrá recibir el registro de teclas presionadas por la víctima. Para hacerlo, se puede utilizar una herramienta como Netcat o Socat.

2.2. Keylogger en Linux.

Una implementación bastante habitual en sistemas Linux es Xorg, el cual utiliza XKB (X Keyboard Extension) para configurar el mapa de caracteres del teclado dependiendo del idioma utilizado por el usuario.

En este caso concreto, la librería "Xlib" provee una función llamada "XQueryKeymap", la cual recibe como parámetro una conexión al servidor X y devuelve un array de bytes con las teclas que se han presionado. Dicho array representa el estado lógico del teclado y se encuentra representado por una matriz de 32 bytes.

Para invocar a dicha función desde un script en Python, existen dos posibilidades, utilizar el módulo "ctypes" para resolver y cargar la librería X11 o instalar y utilizar la librería "python-xlib". En el primer caso, no es necesario tener dependencias instaladas, en el segundo caso, es necesario instalar la librería utilizando "pip" o descargando la última versión disponible (http://sourceforge.net/projects/python-xlib/files/python-xlib/) y ejecutando el script "setup.py".

Si se desea utilizar el módulo "ctypes" para cargar la librería "Xlib", es necesario manejar manualmente los diferentes estados del teclado y su correspondiente mapeo ya que aunque la función "XQueryKeymap" permite acceder al registro de cada una de las teclas presionadas por el usuario, dichos valores no se encuentran en formato ASCII, con lo cual es necesario convertir cada byte a la representación alfanumérica que le corresponde.

El script **7-keyloggerLinux.py** utiliza la librería python-xlib indicada anteriormente, la cual cuenta con clases y funciones que facilitan todo el trabajo. Lo primero que intenta hacer el script anterior es una conexión contra la máquina controlada por el atacante, que en este caso concreto debe tener el puerto 8080 abierto y esperando conexiones.

Por otro lado, una instancia de la clase "Display" permite habilitar un contexto para registrar los eventos de entrada producidos en el servidor X y para gestionar cada uno de dichos eventos, se cuenta con la función "callback". La función "callback" es la que realmente tiene toda la lógica del keylogger, ya que se encarga de validar que se trata de un evento del teclado y posteriormente enviar la información capturada a la máquina controlada por el atacante.

Independientemente de utilizar Xlib directamente con el modulo "ctypes" o la librería "python-xlib", es importante hacer que el script anterior se ejecute en modo silencioso, sin que se levanten ventanas que alerten a la víctima o que simplemente pueda cerrar.

Para hacer esto, existen muchas alternativas, como por ejemplo lanzando el script con la utilidad "nohup" o "screen", distribuyendo este programa junto con las funcionalidades originales de otro programa de confianza para el usuario, etc. Los posibles mecanismos de distribución y ofuscación pueden ser muchos y dependen del objetivo marcado por parte del atacante.

3. PUNTOS CLAVE

- TOR es una red de anonimato que es resistente a la censura y ha sido diseñada con el objetivo de proteger las comunicaciones de aquellas personas que viven en sitios con altos niveles de control y vigilancia.
- Es posible ejecutar peticiones HTTP a sitios en Internet utilizando TOR y requests. La configuración no es muy compleja y solo es necesario indicar en la librería el puerto donde se ejecuta el proxy socks.
- Stem es una librería que ha sido diseñada con el objetivo de administrar una instancia de TOR desde cualquier script en Python. Además de esto, es una muy buena alternativa a la hora de iniciar una instancia de TOR automáticamente y utilizarla desde un script en Python
- En las campañas de Red Team es habitual crear rutinas de espionaje tales como KeyLoggers. Estas rutinas pueden ser utilizadas por atacantes con el objetivo de capturar la interacción completa que tiene el usuario con el sistema.